Developing programs by "Splitting atoms" (rely/guarantee conditions, data reification, . . .)

Cliff B Jones

Computing Science
Newcastle University

FMCO 2008-10-22

Contents

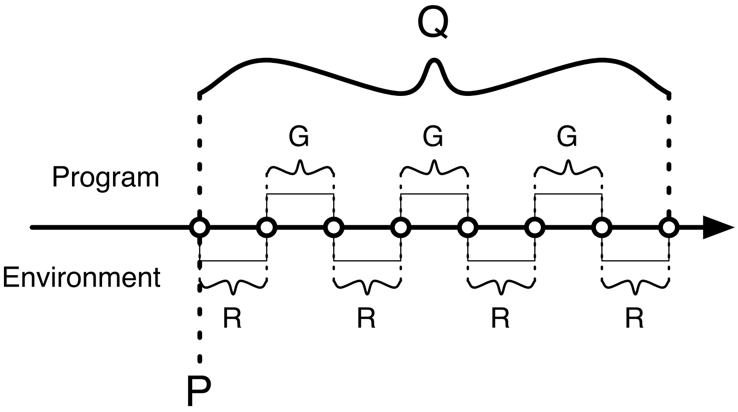
Key abstractions

- Pre/post-conditions (as in VDM/B/. . .)
  - design by *sequential*"operation decomposition rules"
  - Floyd/Hoare-like rules (coping with relational post-conditions)
- Rely/Guarantee "thinking"
  - not (just) a specific set of rules
  - show importance of "frames" (cf. Separation Logic)
  - using "auxiliary variables"
- Abstract objects
  - choice of abstract data objects key for specifications
  - data "reification" (classic-VDM / Nipkow's rule)
  - link with R/G development
- "fiction of atomicity"
  - "splitting (software) atoms safely" [Jon07]
  - cf. database transactions [JLRW05], . . .

While (operation decomposition) rule

$$\boxed{While\text{-}I}\ \frac{\begin{array}{l} S\ \textbf{sat}\ (P \wedge b, P \wedge W) \\ P \ \Rightarrow\ \delta_l(b) \end{array}}{mk\text{-}While(b, S)\ \textbf{sat}\ (P, P \wedge \neg\, b \wedge W^*)}$$

An R/G picture



One R/G rule

cf. [CJ07]

$$
\boxed{Par\text{-}I}\;\frac{\begin{array}{l}\{P, R \lor Gr\} \vdash sl \textbf{ sat } (Gl, Ql)\\ \{P, R \lor Gl\} \vdash sr \textbf{ sat } (Gr, Qr)\\ Gl \lor Gr \;\Rightarrow\; G\\ \overleftarrow{P} \land Ql \land Qr \land (R \lor Gl \lor Gr)^* \;\Rightarrow\; Q\end{array}}{\{P, R\} \vdash mk\text{-}Par(sl, sr) \textbf{ sat } (G, Q)}
$$

Subtle link between R/G and data reification

cf. [Jon07]

- in $FINDP$
  - we have $t \leftarrow min(t, local)$ in n parallel processes
  - assuming we don't want to "lock" t
  - need a representation that helps us to preserve R/G conditions
  - (simple to) represent as t as $min(et, ot)$
- SIEVE
  - we have to remove an element from a set s
  - assuming we don't want to "lock" s (big!)
  - need a representation that helps preserve R/G conditions $s \subseteq \overleftarrow{s}$
  - (less obvious) represent s as a bit vector
- Simpson
  - extremely interesting
  - my claim: this is the essence of Simpson's contribution

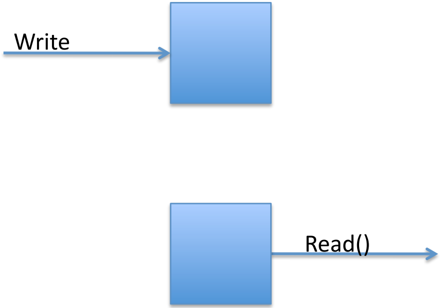
Contents

ACMs: topic of [JP08]

Communication (Atomic?)

ACMs

Atomic and (trying for) Asynchronous



Simpson's algorithm

- Simpson's algorithm
  - ingenious algorithm
  - difficult to prove correct
  - actually, all proofs make assumptions
  - different verification methods give different insights
  - but, even then, lack of *explanation*
- several other folk still working on this
  - come back to at end
- run through our "rational reconstruction"
  - "explanation" via layers of abstraction
- essential to get the big steps right before detailed proof
- apologies for so much argument about eight lines of code ...


...AND THAT, IN A NUTSHELL, IS HOW THIS PROGRAMME WORKS.
Charteris

Specification

$$\Sigma^a :: \ data\text{-}w: Value^*$$
$$fresh\text{-}w: \mathbb{N}$$
$$hold\text{-}r: \mathbb{N}$$

$$\textbf{inv}\ (mk\text{-}\Sigma^a(data\text{-}w, fresh\text{-}w, hold\text{-}r)) \triangleq$$
$$fresh\text{-}w, hold\text{-}r \in \{1..\textbf{len}\ data\text{-}w\} \wedge hold\text{-}r \leq fresh\text{-}w$$

$$\sigma_0^a = mk\text{-}\Sigma^a([\mathtt{x}], 1, 1)$$

while true do
$\quad start\text{-}Write(v: Value): data\text{-}w \leftarrow data\text{-}w \frown [v];$
$\quad commit\text{-}Write(): fresh\text{-}w \leftarrow \textbf{len}\ data\text{-}w$
od
while true do
$\quad start\text{-}Read(): hold\text{-}r \leftarrow fresh\text{-}w;$
$\quad end\text{-}Read()r: Value: r \leftarrow data\text{-}w(i)$ **for some** $i \in \{hold\text{-}r..fresh\text{-}w\}$
od

Examples 1, 2

$$
\begin{array}{lcl}
\mathit{start\text{-}Write}(\mathrm{y}) & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 1, 1) \\
\mathit{commit\text{-}Write}() & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 2, 1) \\
\mathit{start\text{-}Read}() & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 2, 2) \\
\mathit{end\text{-}Read}() & .. & r = \mathrm{y}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{start\text{-}Write}(\mathrm{y}) & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 1, 1) \\
\mathit{start\text{-}Read}() & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 1, 1) \\
\mathit{end\text{-}Read}() & .. & r = \mathrm{x} \\
\mathit{commit\text{-}Write}() & .. & \mathit{mk}\text{-}\Sigma^a([\mathrm{x}, \mathrm{y}], 2, 1)
\end{array}
$$

Example 3

$$
\begin{array}{lcl}
\textit{start-Read}() & .. & \textit{mk-}\Sigma^a([\mathtt{x}], 1, 1) \\
\textit{start-Write}(\mathtt{y}) & .. & \textit{mk-}\Sigma^a([\mathtt{x}, \mathtt{y}], 1, 1) \\
\textit{commit-Write}() & .. & \textit{mk-}\Sigma^a([\mathtt{x}, \mathtt{y}], 2, 1) \\
\textit{start-Write}(\mathtt{z}) & .. & \textit{mk-}\Sigma^a([\mathtt{x}, \mathtt{y}, \mathtt{z}], 2, 1) \\
\textit{commit-Write}() & .. & \textit{mk-}\Sigma^a([\mathtt{x}, \mathtt{y}, \mathtt{z}], 3, 1) \\
\textit{end-Read}() & .. & r \in \{\mathtt{x}, \mathtt{y}, \mathtt{z}\} \\
\textit{start-Read}() & .. & \textit{mk-}\Sigma^a([\mathtt{x}, \mathtt{y}, \mathtt{z}], 3, 3) \\
\textit{end-Read}() & .. & r = \mathtt{z}
\end{array}
$$

Specification in terms of four sub-operations ($Write$)

Atomic operations — therefore pure pre/post specification

while true do
  $start\text{-}Write(v\colon Value)\colon data\text{-}w \leftarrow data\text{-}w \curvearrowright [v];$
  $commit\text{-}Write()\colon fresh\text{-}w \leftarrow$ **len** $data\text{-}w$
od
$||$
$\vdots$

$Write(v\colon Value)$
  $start\text{-}Write(v\colon Value)$
    wr $data\text{-}w$
    post $data\text{-}w = \overleftarrow{data\text{-}w} \curvearrowright [v]$
  $commit\text{-}Write(v\colon Value)$
    rd $data\text{-}w$
    wr $fresh\text{-}w$
    pre $data\text{-}w($**len** $data\text{-}w) = v$
    post $fresh\text{-}w =$ **len** $data\text{-}w$

Specification in terms of four sub-operations ($Read$)

$\vdots$

$||$

while true do

$\quad start\text{-}Read()\colon hold\text{-}r \leftarrow fresh\text{-}w;$

$\quad end\text{-}Read()r\colon Value\colon r \leftarrow data\text{-}w(i)$ **for some** $i \in \{hold\text{-}r..fresh\text{-}w\}$

od

$Read()r\colon Value$

local $hold\text{-}r\colon \mathbb{N}$

$start\text{-}Read()$

wr $hold\text{-}r$

rd $fresh\text{-}w$

post $hold\text{-}r = fresh\text{-}w$

$end\text{-}Read()r\colon Value$

rd $data\text{-}w, fresh\text{-}w$

post $\exists i \in \{hold\text{-}r..fresh\text{-}w\} \cdot r = data\text{-}w(i)$

General messages

- note “algorithmic” specification
- “fiction of atomicity”
  - but single “atomic” variable does not cover all behaviour
- “frames” (for rd/wr access)
  - plus “local”
- data abstraction

Splitting atoms in Σ^a ($Write$)

Accept overlap (only read/write) — therefore rely/guarantee

$Write(v: Value)$

$start\text{-}Write(v: Value)$

rd $fresh\text{-}w$

wr $data\text{-}w$

rely $fresh\text{-}w = \overleftarrow{fresh\text{-}w} \wedge data\text{-}w = \overleftarrow{data\text{-}w}$

guar $\{1..fresh\text{-}w\} \lhd data\text{-}w = \{1..fresh\text{-}w\} \lhd \overleftarrow{data\text{-}w}$

post $data\text{-}w = \overleftarrow{data\text{-}w} \curvearrowright [v]$

$commit\text{-}Write(v: Value)$

rd $data\text{-}w$

wr $fresh\text{-}w$

pre $data\text{-}w(\mathbf{len}\ data\text{-}w) = v$

rely $fresh\text{-}w = \overleftarrow{fresh\text{-}w} \wedge data\text{-}w = \overleftarrow{data\text{-}w}$

post $fresh\text{-}w = \mathbf{len}\ data\text{-}w$

Splitting atoms in Σ^a $(Read)$

$$
\begin{array}{l}
Read()r\colon Value \\
\quad start\text{-}Read() \\
\qquad \textbf{rd}\ fresh\text{-}w \\
\qquad \textbf{wr}\ hold\text{-}r \\
\qquad \textbf{rely}\ hold\text{-}r = \overleftarrow{hold\text{-}r} \\
\qquad \textbf{post}\ hold\text{-}r \in \{\overleftarrow{fresh\text{-}w}, fresh\text{-}w\} \\
\quad end\text{-}Read()r\colon Value \\
\qquad \textbf{rd}\ data\text{-}w, fresh\text{-}w, hold\text{-}r \\
\qquad \textbf{rely}\ hold\text{-}r = \overleftarrow{hold\text{-}r} \wedge \forall i \in \{hold\text{-}r .. \overleftarrow{fresh\text{-}w}\} \cdot data\text{-}w(i) = \overleftarrow{data\text{-}w}(i) \\
\qquad \textbf{post}\ \exists i \in \{hold\text{-}r .. \overleftarrow{fresh\text{-}w}\} \cdot r = \overleftarrow{data\text{-}w}(i)
\end{array}
$$

General messages

- phasing
  - makes clear $start\text{-}Write$ cannot interfere with $commit\text{-}Write$
  - avoids implications in rely conditions
- frames plus phasing significantly simplify R/G assertions
- cf. $rely\text{-}start\text{-}Write$ on Σ^a above

Retaining less history

A data reification exercise — still very general

$$
\begin{aligned}
\Sigma^i \;::\; & \mathit{data\text{-}w}\colon X \xrightarrow{m} \mathit{Value} \\
& \mathit{fresh\text{-}w}\colon X \\
& \mathit{hold\text{-}r}\colon X \\
& \mathit{hold\text{-}w}\colon X
\end{aligned}
$$

$$
\textbf{inv}\ (mk\text{-}\Sigma^i(\mathit{data}, \mathit{fresh}, \mathit{hold\text{-}r}, \mathit{hold\text{-}w})) \triangleq \{\mathit{fresh}, \mathit{hold\text{-}r}, \mathit{hold\text{-}w}\} \subseteq \textbf{dom}\ \mathit{data}
$$

$$
\sigma_0^i = mk\text{-}\Sigma^i(\{\alpha \mapsto \mathtt{x}\}, \alpha, \alpha, \alpha)
$$

Relating Σ^i to Σ^a

Using Nipkow's rule

$$r(\sigma_1^a, \sigma_1^i) \wedge post^i(\sigma_1^i, \sigma_2^i) \;\Rightarrow\; \exists \sigma_2^a \in \Sigma^a \cdot post^a(\sigma_1^a, \sigma_2^a) \wedge r(\sigma_2^a, \sigma_2^i)$$

$$r : \Sigma^a \times \Sigma^i \to \mathbb{B}$$

$$\begin{array}{l} r(mk\text{-}\Sigma^a(data\text{-}w^a, fresh\text{-}w^a, hold\text{-}r^a), \\ \qquad\qquad mk\text{-}\Sigma^i(data\text{-}w^i, fresh\text{-}w^i, hold\text{-}r^i, hold\text{-}w^i)) \quad \triangleq \\ \quad \textbf{rng}\ data\text{-}w^i \subseteq \textbf{elems}\ data\text{-}w^a \wedge \\ \quad data\text{-}w^a(fresh\text{-}w^a) = data\text{-}w^i(fresh\text{-}w^i) \wedge \\ \quad data\text{-}w^a(hold\text{-}r^a) = data\text{-}w^i(hold\text{-}r^i) \end{array}$$

Specifications of the sub-operations on Σ^i

Still overlapped — still rely/guarantee

$Write(v\colon Value)$
  local $hold\text{-}w\colon X$
  $start\text{-}Write(v\colon Value)$
    rd $hold\text{-}r, fresh\text{-}w$
    wr $data\text{-}w, hold\text{-}w$
    rely $fresh\text{-}w = \overleftarrow{fresh\text{-}w} \wedge data\text{-}w = \overleftarrow{data\text{-}w}$
    guar $\{\overleftarrow{hold\text{-}r}, hold\text{-}r\} \lhd data\text{-}w = \{\overleftarrow{hold\text{-}r}, hold\text{-}r\} \lhd \overleftarrow{data\text{-}w}$
    post $hold\text{-}w \in (X - \{fresh\text{-}w, \overleftarrow{hold\text{-}r}, hold\text{-}r\}) \wedge data\text{-}w = \overleftarrow{data\text{-}w} \dagger \{hold\text{-}w \mapsto v\}$
  $end\text{-}Write(v\colon Value)$
    rd $data\text{-}w, hold\text{-}w$
    wr $fresh\text{-}w$
    pre $data\text{-}w(hold\text{-}w) = v$
    rely $fresh\text{-}w = \overleftarrow{fresh\text{-}w} \wedge data\text{-}w = \overleftarrow{data\text{-}w}$
    post $fresh\text{-}w = hold\text{-}w$

Specifications of the sub-operations on Σ^i

$Read()r \colon Value$
  $start\text{-}Read()$
    rd $fresh\text{-}w$
    wr $hold\text{-}r$
    rely $hold\text{-}r = \overleftarrow{hold\text{-}r}$
    post $hold\text{-}r \in \{\overleftarrow{fresh\text{-}w}, fresh\text{-}w\}$
  $end\text{-}Read()r \colon Value$
    rd $hold\text{-}r, data\text{-}w$
    rely $hold\text{-}r = \overleftarrow{hold\text{-}r} \wedge data\text{-}w(hold\text{-}r) = \overleftarrow{data\text{-}w}(hold\text{-}r)$
    post $r = data\text{-}w(hold\text{-}r)$

General messages

- simpler R/G because of read/write frames
- data reification
  - (potentially) reducing non-determinism
  - use of VDM's other reification rule
- still have "bold" atomicity assumptions
  - couldn't update $data\text{-}w$ atomically on any reasonable machine
- still work to be done
- role of data reification in achieving rely conditions
- Simpson's representation crucial

The four-slot representation

Focus on Simpson's inspiration

$$\begin{array}{lll} \Sigma^r & :: & \mathit{data\text{-}w}\colon P \times S \xrightarrow{m} \mathit{Value} \\ & & \mathit{pair\text{-}w}\colon P \\ & & \mathit{pair\text{-}r}\colon P \\ & & \mathit{slot\text{-}w}\colon P \xrightarrow{m} S \\ & & \mathit{wp\text{-}w}\colon P \\ & & \mathit{ws\text{-}w}\colon S \\ & & \mathit{rs\text{-}r}\colon S \end{array}$$

where (key assumptions about granularity (ρ)):

$$P, S = \mathit{Token}\text{-}\mathbf{set}$$

$$P = S$$
$$\mathbf{card}\ P = 2$$
$$\rho(i) \neq i$$

Connection Σ^r with Σ^i

Σ^i	represented in Σ^r by
$data\text{-}w^i$	$data\text{-}w^r$
$fresh\text{-}w^i$	$(pair\text{-}w^r, slot\text{-}w^r(pair\text{-}w^r))$
$hold\text{-}r^i$	$(pair\text{-}r^r, slot\text{-}w^r(pair\text{-}r^r))$
$hold\text{-}w^i$	$(wp\text{-}w^r, wp\text{-}s^r)$

Specifications of the sub-operations on Σ^r

$Write(v \colon Value)$
  local $wp\text{-}w \colon P$
  local $ws\text{-}w \colon S$
  $start\text{-}Write(v \colon Value)$
    rd $pair\text{-}r,\ slot\text{-}w$
    wr $data\text{-}w$
    rely $slot\text{-}w = \overleftarrow{slot\text{-}w} \wedge data\text{-}w = \overleftarrow{data\text{-}w}$
    guar $\{(\overleftarrow{pair\text{-}r}, slot\text{-}w(\overleftarrow{pair\text{-}r})), (pair\text{-}r, slot\text{-}w(pair\text{-}r))\} \lhd data\text{-}w =$
$\{(\overleftarrow{pair\text{-}r}, slot\text{-}w(\overleftarrow{pair\text{-}r})), (pair\text{-}r, slot\text{-}w(pair\text{-}r))\} \lhd \overleftarrow{data\text{-}w}$
    post $wp\text{-}w = \rho(\overleftarrow{pair\text{-}r}) \wedge ws\text{-}w = \rho(slot\text{-}w(wp\text{-}w)) \wedge data\text{-}w(wp\text{-}w, ws\text{-}w) = v$
  $end\text{-}Write()$
    wr $pair\text{-}w,\ slot\text{-}w$
    rely $pair\text{-}w = \overleftarrow{pair\text{-}w} \wedge slot\text{-}w = \overleftarrow{slot\text{-}w}$
    guar $slot\text{-}w(pair\text{-}r) = \overleftarrow{slot\text{-}w}(pair\text{-}r)$
    post $slot\text{-}w(wp\text{-}w) = ws\text{-}w \wedge pair\text{-}w = wp\text{-}w$

Specifications of the sub-operations on Σ^r

$Read()r\colon Value$
  local $rs\text{-}r\colon S$
  $start\text{-}Read()$
    rd $pair\text{-}w, slot\text{-}w$
    wr $pair\text{-}r$
    rely $slot\text{-}w(pair\text{-}r) = \overleftarrow{slot\text{-}w}(pair\text{-}r) \wedge pair\text{-}r = \overleftarrow{pair\text{-}r}$
    post $pair\text{-}r = \overleftarrow{pair\text{-}w} \wedge rs\text{-}r = \overleftarrow{slot\text{-}w}(pair\text{-}r)$
  $end\text{-}Read()r\colon Value$
    rd $pair\text{-}r, data\text{-}w$
    rely $pair\text{-}r = \overleftarrow{pair\text{-}r} \wedge data\text{-}w(pair\text{-}r, rs\text{-}r) = \overleftarrow{data\text{-}w}(pair\text{-}r, rs\text{-}r)$
    post $r = data\text{-}w(pair\text{-}r, rs\text{-}r)$

Satisfies guarantee conditions (as well as post)

$Write(v: Value)$
  local $wp\text{-}w: P$
  local $ws\text{-}w: S$
    $wp\text{-}w \leftarrow \rho(pair\text{-}r);$
    $ws\text{-}w \leftarrow \rho(slot\text{-}w(wp\text{-}w));$
    $data\text{-}w(wp\text{-}w, ws\text{-}w) \leftarrow v;$
    $slot\text{-}w(wp\text{-}w) \leftarrow ws\text{-}w;$
    $pair\text{-}w \leftarrow wp\text{-}w$

$Read()r: Value$
  local $rs\text{-}r: S$
    $pair\text{-}r \leftarrow pair\text{-}w;$
    $rs\text{-}r \leftarrow slot\text{-}w(pair\text{-}r);$
    $r \leftarrow data\text{-}w(pair\text{-}r, rs\text{-}r)$

Contents

Comparisons

- Henderson's thesis (JSF/CBJ supervision)
  - use "shrinking sequence" in specification
  - different approaches (including CSP/FDR) highlight facets
  - up to, including "meta-stability" of control bits
- event refinement (Abrial)
  - f/g to "avoid" algorithmic specification
  - we were working on proof — in communication
  - non-deterministic order of events, virtual "instruction counter"
  - refine one event to many: all but one "refines skip"
- Separation Logic (Bornat, Parkinson, Vafeiadis, O'Hearn)
  - "frame" defined by alphabet of assertions
  - notation certainly more compact
  - expected it to be much better on 4-slot because of "ownership"
  - in fact, doesn't offer intuition

Conclusions

- all at FMCO probably accept "refinement from abstractions"
- "splitting atoms" – a new/old formal addition
- subsidiary points
  - rely/guarantee "thinking"
  - remember frame descriptions
  - combination with data reification
  - link with "phasing"
  - "auxiliary variables" + Nipkow's rule
  - ...
  - tool support (ASE*2)
- one further technical issue
  - expressiveness of R/G (thanks to Viktor Vafeiadis)

References

J. W. Coleman and C. B. Jones.
A structural proof of the soundness of rely/guarantee rules.
Journal of Logic and Computation, 17(4):807–841, 2007.

C. B. Jones, D. Lomet, A. Romanovsky, and G. Weikum.
The atomicity manifesto.
Journal of Universal Computer Science, 11(5):636–650, 2005.

C. B. Jones.
Splitting atoms safely.
Theoretical Computer Science, 357:109–119, 2007.

Cliff B. Jones and Ken G. Pierce.
Splitting atoms with rely/guarantee conditions coupled with data reification.
In *ABZ2008*, volume LNCS 5238, pages 360–377, 2008.